This paper presents implementation and optimization of air quality model CAMx using Cuda C targeting GPU clusters. Experiment results show that GPU-HADVPPM can achieve about 1000x acceleration, the series of optimization can acheive about dozens of times acceleration and the final version of CAMx-GPU can achieve 4.5x speedup with 8 CPU cores and 8 GPU accelerators on V100 cluster. Here are some specific comments.

Response: We appreciate the editor for reviewing our manuscript and for the valuable suggestions, which we will address point by point in the following.

1. This paper need to be more convincing about how some experiment results will be explained. Such as: the offline experiment achieves about 1000x acceleration which is far exceed the ratio of theoretical peak performance of GPU and CPU.

Response: Thanks for the constructive comment. In the thread hierarchy, the thread is the most basic unit of GPU for parallel computing. Threads can be organized into one-dimensional, two-dimensional, or three-dimensional blocks. In a three-dimensional block, each dimension can contain a maximum number of threads of 1024, 1024, and 64, respectively. Similarly, blocks can be organized into one-dimensional, two-dimensional, or three-dimensional grids. In a three-dimensional grid, each dimension can contain a maximum number of blocks of $2^{31}-1$, 65535, and 65535, respectively. It is theoretically possible to distribute a large number of copies of kernel functions into tens of billions of threads for parallel computing without exceeding the GPU memory. In the offline performance test, we adopted a one-dimensional thread organization, with 1024 threads per block and several blocks based on the array size. For example, in the data size of 10^7, the GPU can implement the parallel computation of 10 million threads, while the CPU can only use serial cyclic computation. Therefore, the offline experiments achieve up to 1100x acceleration, and the larger the data size, the more pronounced the advantage of GPU multithreading architecture and the higher the computing efficiency. In addition, the data transfer time between CPU and GPU is not considered in the offline performance experiments. Also, without I/O, the GPU-based SBU-YLIN scheme in the WRF model can achieve **896x speedup** compared to a Fortran implementation running on a CPU (Mielikainen et al.,2012). We have revised this part in lines 463-474, which are as follows:

As described in Sect. 3.2, the thread is the most basic unit of GPU for parallel computing. Each dimension of the three-dimensional block can contain a maximum number of threads of 1024,1024, and 64, respectively. Each dimension of the three-dimensional grid can contain a maximum number of blocks of $2^{31}-1$, 65535, and 65535. It is theoretically possible to distribute a large number of copies of kernel functions into tens of billions of threads for parallel computing without exceeding the GPU memory. In the offline performance experiments, the GPU achieved up to 10 million threads of parallel computing, while the CPU can only use serial cyclic computation. Therefore, GPU-HADVPPM achieves a maximum acceleration of about 1100x without I/O. In addition to this study, the GPU-based SBU-YLIN scheme in the WRF model can achieve 896x acceleration compared to the Fortran implementation running on the CPU (Mielikainen et al., 2012b).

2. It is suggested to supplement performance data of CAMx implemented by Fortran and C in section4.4.1.

Response: Thanks for the constructive comment. We supplement the coupling performance experiment results for the C version of HADVPPM. On the K40m and V100 cluster, the elapsed time of the C version HADVPPM is 51.4 seconds and 45.2 seconds, respectively, 26.7% and 33.4% lower than that of the Fortran version HADVPPM. We have revised this part in lines 519-534, which are as follows:

In terms of the single module computational efficiency of HADVPPM and GPU-HADVPPM, we further coupling test the computational performance of the Fortran version HADVPPM on the CPU, C version HADVPPM on the CPU, and CUDA C version GPU-HADVPPM in CAMx-CUDA V1.4 (GPU-HADVPPM V1.4) on the GPU, using system_clock functions in the Fortran language and cudaEvent_t in CUDA programming. The specific results are shown in Figure 11. On the K40m cluster, it takes 37.7 seconds and 51.4 seconds to launch the Intel Xeon E5-2682 v4 CPU to run Fortran and C version HADVPPM, the C version is 26.7% slower than the Fortran version. After the CUDA technology was used to convert the C code into CUDA C, the CUDA C version took 29.6 seconds to launch an NVIDIA Telsa K40m GPU to run GPU-HADVPPM V1.4, with 1.3x and 1.7x acceleration. On the V100 cluster, the Fortran, the C, and the CUDA C version are computationally more efficient than those on the K40m cluster. It takes 30.1 seconds and 45.2 seconds to launch Intel Xeon Platinum 8168 CPU to run Fortran and C version HADVPPM and 1.6

seconds to run the GPU-HADVPPM V1.4 using an NVIDIA V100 GPU. The computational efficiency of the CUDA C version is 18.8x and 28.3x higher than Fortran and C versions.

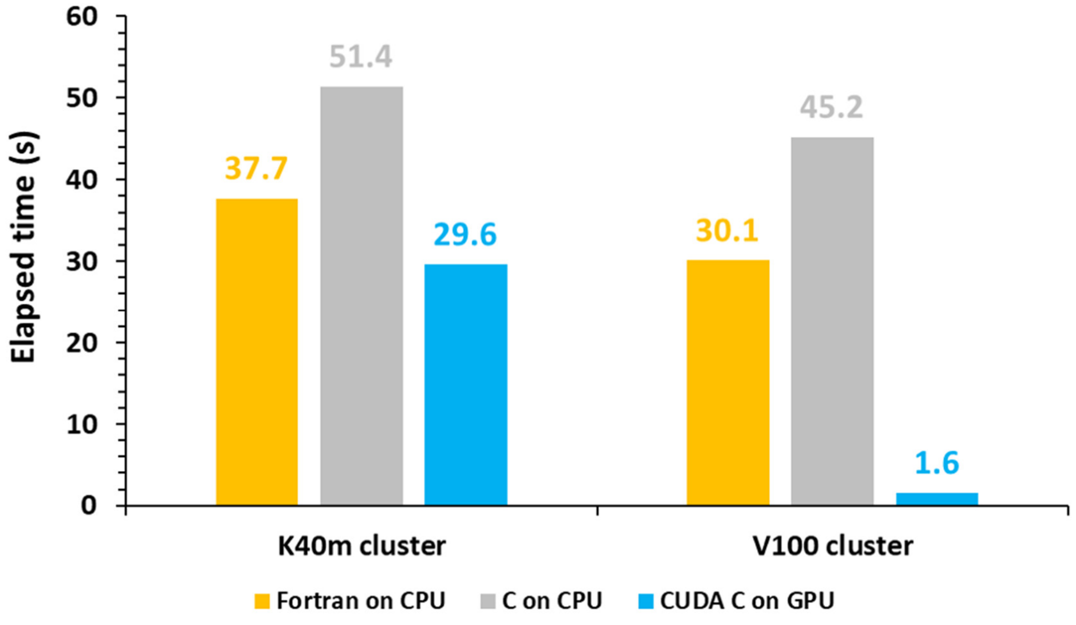

Figure 11. The elapsed time of the Fortran version HADVPPM on the CPU, the C version HADVPPM on the CPU, and CUDA C version GPU-HADVPPM V1.4 on the GPU. The unit is seconds (s).

3. There are some typing errors, such as CAS-EMS should be CAS-ESM, CAMx-V1.0 in line 492 should be CAMx-CUDA V1.0.

Response: Sorry for this mistake. We have corrected typing errors in line 64, 84, and 504, which are as follows:

Line 64: The GPU has proven successful in weather models such as Non-Hydrostatic Icosahedral Model (NIM; Govett et al.,2017), Global/Regional Assimilation and Prediction System (GRAPES; Xiao et al., 2022), and Weather Research and Forecasting model (WRF; Huang et al., 2011; Huang et al., 2012; Mielikainen et al., 2012a; Mielikainen et al., 2012b; Mielikainen et al., 2013a ; Mielikainen et al., 2013b; Price et al., 2014; Huang et al., 2015), ocean models such as LASG/IAP Climate System Ocean Model (LICOM; Jiang et al., 2019; Wang et al., 2021a) and Princeton Ocean Model (POM; Xu et al., 2015), and the Earth System Model of Chinese Academy of Sciences (**CAS-ESM**; Wang et al., 2021b ; Wang et al., 2021c).

Line 84: In terms of climate system model, Wang et al., (2021c) and Wang et al., (2021b) used CUDA Fortran and CUDA C to carry out heterogeneous porting of the RRTMG_SW and RRTMG_LW scheme of the atmospheric component model of the **CAS-ESM** earth system model, and achieved a 38.88x and 77.78x acceleration respectively.

Line 504: In CAMx-CUDA V1.2, the frequency of data transmission between CPU-GPU within one time step is reduced to 1, and the elapsed time on the two heterogeneous clusters is 1207 seconds and 548 seconds, respectively, and the speedup is 9.0x and 68.0x compared to the **CAMx-CUDA V1.0**.

Reference

Mielikainen, J., Huang, B., Huang, H.-L. A., and Goldberg, M. D.: GPU Implementation of Stony Brook University 5-Class Cloud Microphysics Scheme in the WRF, IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 5, 625-633, 10.1109/jstars.2011.2175707, 2012.